Compression of TPC data in the ALICE experiment ⋆

A. Nicolaucig [a,*], M. Mattavelli [a], S. Carrato [b]

[a] *Integrated System Laboratory (LSI/ISL), Swiss Federal Institute of Technology, CH-1015 Lausanne, Switzerland*

[b] *Image Processing Laboratory, D. E. E. I., University of Trieste, via A. Valerio 10, I-34127 Trieste, Italy*

Abstract

In this paper two algorithms for the compression of the data generated by the Time Projection Chamber (TPC) detector of the ALICE experiment at CERN are described. The first algorithm is based on a lossless source code modeling technique, i.e. the original TPC signal information can be reconstructed without errors at the decompression stage. The source model exploits the temporal correlation that is present in the TPC data to reduce the entropy of the source. The second algorithm is based on a source model which is lossy if samples of the TPC signal are considered one by one. Conversely, the source model is lossless or quasi-lossless if some physical quantities that are of main interest for the experiment are considered. These quantities are the area and the location of the center of mass of each TPC signal pulse.

Obviously entropy coding is applied to the set of events defined by the two source models to reduce the bit rate to the corresponding source entropy. Using TPC simulated data according to the expected ALICE TPC performance, the lossless and the lossy compression algorithms achieve a data reduction respectively to 49.2% and in the range of 34.2% down to 23.7% of the original data rate. The number of operations per input symbol required to implement the compression stage for both algorithms is relatively low, so that a real-time implementation embedded in the TPC data acquisition chain using low–cost integrated electronics is a realistic option to effectively reduce the data storing cost of ALICE experiment.

Key words: ALICE, TPC, compression, lossless, lossy, arithmetic coding
PACS: 29.40.Gx
1991 MSC: 94A29, 94A45

Introduction

ALICE (A Large Ion Collider Experiment) is an experiment that will be held in 2005 at the LHC (Large Hadron Collider) at CERN [1,2]. The experiment will study collisions between heavy ions with energies around 5.5 TeV. The collisions will take place at the center of a set of several detectors, which are designed to track and identify the particles produced.

One of the main detectors of the ALICE experiment is the Time Projection Chamber (TPC). The TPC is a large horizontal cylinder, filled with gas, where a suitable axial electric field is present. When particles pass through, they ionize the gas atoms, and the resulting electrons drift in the electric field. By measuring the arrival of electrons at the end of the chamber, the TPC can reconstruct the path of the original charged particles. The electrons are collected by more than 570 000 sensitive pads where the signal is acquired in the form of pulses, each corresponding to the passage of one particle. This signal is amplified by a preamplifier/shaper and digitalized by a 10 bit A/D converter at a sampling frequency of 5.66 MHz. The digitalized signal is processed and formatted by an Application Specific Integrated Circuit (ASIC) called ALTRO (ALICE TPC ReadOut). At this stage, the overall throughput of the 570 000 channels is around 8.4 GByte/s.

Considering that the duration of the experiment is forecasted in a few months time lap, it is clear that the amount of data to be collected is expected to be extremely large. So as to keep the complexity and cost of the data storage equipment as low as possible, the goal is to reduce the volume of data using suitable data compression methods. The cost reduction of the data storage system can be considered roughly proportional to the data compression factor. It is advisable to implement the compression system in the front-end electronics at the output of the ALTRO circuit, so that the cost for the optical links, which carry data out of the chamber to the following stages of the acquisition chain, is also reduced [3].

Compression techniques can be classified into lossless and lossy depending on how the model of the information source defines, or better *models*, the set of events that are then entropy coded. Using a lossless source model the data can be exactly reconstructed as they were before compression. The use of a lossy source model, justified by the fact that it generally can provide significantly higher compression ratios compared to lossless models, has the

⋆ Authors would like to thank Luciano Musa, Karel Safarik, Johanna Stachel, Jouri Belikov, Vincent Colin de Verdiere, Boris Breidenbach and Marian Ivanov (CERN) for the help on the ALICE experiment and for the provided simulated data.
* Corresponding author. Telephone: +41 21 693 6982. Fax: +41 21 693 4663.
Email address: Aldo.Nicolaucig@epfl.ch (A. Nicolaucig).

drawback that an error in the reconstruction of data must be accepted. Lossy source models have become very popular in the last decade in the field of audio and video compression for their remarkable performance [**?**]. Lossy models have been carefully designed so that reconstruction errors are not perceived using psycovisual or psycoacoustic models or they remain comparable with the intrinsic signal noise.

In this paper, after a brief description of the format of the TPC data that are to be processed, first a lossless algorithm which exploits the temporal correlation among the samples is described. Then, a rigorously speaking lossy source model, in which however some quantities of physical interest such as the energy (area of the electrical pulse) and the temporal position of each pulse (center of mass of the pulse) registered by the TPC pads are exactly preserved, is described. Such physical quantities carried by the signal are the ones of interest for the overall ALICE experiment results [1], i.e. the reconstruction of the particle trajectories. The performances of the two algorithms are reported, and the corresponding computational complexity is discussed, aiming at evaluating a possible implementation of the system on low-cost electronic devices.

1 The ALICE TPC read out data format

Before describing the compression algorithm, it is necessary to spend a few words on the format of data at the output of ALTRO circuit so as to understand how the compression algorithms are applied; these data are indeed the input of the compression system [1].

In the so called ALTRO data format, only the samples over a given threshold are considered, while the others are discarded. This means that, if we call *bunch* a group of adjacent over-threshold samples coming from one pad, the signal can be represented "bunch by bunch". More precisely, a bunch is described by three fields: temporal information (temporal position of the last sample in the bunch, one 10 bit word [1]), bunch length (i.e., number of samples [2] in the bunch, one 10 bit word), and sample amplitude values (10 bit words, i.e. a range between 0 and 1023).

[1] It may be noted that, for each trigger selected collision, the acquisition process completes in 88 μs, which implies, at a sampling frequency of 5.66 MHz, a range for time information between 0 and 499, so that one 10 bit word suffices.

[2] Actually, the value transmitted by the ALTRO circuit is the number of samples *plus one* [1].

2 Lossless compression of TPC signals

The lossless technique described in this paper is mainly based on an appropriate probability model for each data field of the ALTRO data format. More precisely, specific probability models for each sample in a bunch are developed. Such models intend to capture both temporal correlation among samples and the characteristic shape of TPC electrical pulses. For what concerns the time information, i.e position of the bunches they are not represented as an absolute value, like in the ALTRO data format, but they are differentially coded using the number of zero samples preceding the bunch. Finally, the bunch length is directly entropy coded.

2.1 Sample values coding

This subsection describes a first basic model, and then introduces a more sophisticated one that can provide higher performances in terms of compression efficiency.

2.1.1 Coding model based on the sample position

Data compression can be obtained by directly applying entropy coding to the sample values without any modeling of the information source (this method will be referred to as EC, Entropy Coding, in Sec. 4). However, improvements in compression performance can be obtained by appropriate modeling. A first improvement has been achieved by observing that the statistics of the signal sample values depends on the position of the sample itself in the bunch. For example, Fig. 1 shows a short interval of a TPC signal on a pad, according to [4]; the signal has been obtained by Montecarlo simulations of the TPC behavior. Observing the signal, it is clear that, due to the pseudo Gaussian shape of most of the bunches, the first and the last sample of each bunch are likely to have a smaller value with respect to those in central positions. Similarly, small values are also expected for isolated samples, i.e. belonging to one-sample bunches. Therefore, it resulted to be useful to classify samples into three classes: one class for isolated samples, one for samples at the beginning or at the end of multiple sample bunches, and one for samples in the central positions of a bunch. Such classification is schematically shown in figure 2, where dots represent samples, and rows relate to bunches classified by length, (i.e. the first row represents one sample bunches, the second row two sample bunches, etc.); the regions represent the classes. By using three different probability distributions for entropy coding, each class results optimized versus the statistics of the corresponding event (shown in Fig. 3); thus, sample values can

be coded more efficiently than using only one probability distribution. This coding scheme will be referred to as SP, Sample Position, in Sec. 4.

2.1.2 Source models exploiting temporal correlation

Generally, an improvement on compression performances can be expected by also exploiting temporal correlation, i.e. the correlation between consecutive samples; this can be done by implementing a suitable prediction scheme. For TPC data, experiments has shown that a simple correlation between adjacent samples does not provide an efficient prediction of the sample values [5]. However, a more sophisticated scheme which captures sample correlation can be derived achieving a further reduction of the overall bit rate. More precisely, with reference to Fig. 1, it can be observed that large samples are likely to be followed by samples of the same value range (and, conversely, for small samples), so that, for each sample, it may be advisable to choose a different code according to the range of values of the preceding one.

More in detail, the developed approach is explained with an example in figure 4, where a three sample bunch is considered. Let us suppose that the first two samples have already been coded and that the third one has to be coded. The code to be used for sample No. 3 may be chosen among eight possible codes according to the value of sample No. 2. In particular, this is done by subdividing the range of sample No. 2 (i.e. $0 \ldots 1023$) into 8 intervals, and associating a different code (for the third sample) to each of these intervals.

It may be noted that, in general, small value samples are more likely to occur than large values; consequently, so as to optimize the use of the 8 codes, it is not advisable to use intervals having the same length, because in this case codes related to large values would be very rarely used. On the contrary, the range $0 \ldots 1023$ is partitioned in such a way that the probability for the sample No. 2 to fall into each interval be the same; this leads to have shorter intervals in the low amplitude side of the available range. In Fig. 5 the 8 possible distributions of sample 3 are shown (again with reference to the simulated data [4]) according to the interval which sample 2 belongs to.

This conditioned probability model can be extended to all the samples that are not in the first position in the bunch, for every bunch length. However, if real-time implementation constraints are taken into account, and in particular the need to reduce the memory size of the model, it is not advisable to have an exceedingly large number of codes. Consequently, samples are partitioned into only four classes, as shown in figure 6, to keep the complexity of the model low. This limitation does reduce the efficiency of the model but the reduction is only of the order of 0.6% [5]. Class 1 and 2 are dedicated to isolated samples and to samples that are at the beginning of bunches, respectively; a simple probability

model is applied to each of these classes, because obviously the use of multiple codes as mentioned above does not apply. Class 3 is for samples in intermediate position in the bunches, and class 4 for samples at the end of bunches; samples in these classes are coded exploiting temporal correlation with the conditioned probability model described above. Eight symbol frequency tables are used for each of these two classes.

This coding scheme will be referred to as TC, Temporal Correlation, in Sec. 4.

2.2 Time information

As already mentioned, in the so called ALTRO data format time information is represented as the 10 bit cardinal number of the time-bin of the last sample of the bunch. This representation can be easily and effectively substituted by the distance between two consecutive bunches, i.e. the number of zero samples between the bunches. Entropy coding can then be applied to this distances.

2.3 Bunch length

In the ALTRO data format, the bunch length is represented as a 10 bit number of samples in the bunch.

Since no apparent correlation with other data (e.g., length of adjacent bunches) exists and no better model (i.e. a model of events with lower entropy) could be found, this information has been coded directly using the simple probability distribution.

2.4 Other lossless coding models

In this subsection, other coding models that have been investigated, but have not provided sensible improvement for lossless compression of the TPC data, are briefly described.

Space correlation

In the trial to exploit space correlation, two models have been considered. The first is based on spatially conditioned probability, the second on a predictive model.

The former is the equivalent, in the spatial domain, to what has been done for time correlation. More precisely, different codes are available to code the samples; for each sample, the appropriate conditioning is selected according to the value of the samples in the same time-bin but in adjacent pads. However, this method resulted in providing poorer performances when compared with the one which exploits time correlation (the comparison being done using the same model complexity, i.e. number of probability distributions available in memory). Moreover, these two techniques can not be easily combined, i.e. it is not possible to exploit both temporal *and* spatial correlation at the same time, because this would require a very large number of probability distributions (i.e. code tables).

The second method that has been investigated consists in predicting the sample values from the values of the samples in adjacent pads and coding the error of this prediction; unfortunately, also for this model performances resulted not to be very good.

It has also been noted that pulses on one channel often resemble temporally shifted versions of those in adjacent channels; then, with the aim of improving performances, both the two methods described above have been modified by adding a first stage which shifts pulses so as to increase spatial correlation with adjacent channels. However, although performances have slightly improved, the increase in efficiency was much lower than expected.

From these experimental results it appears that it is not simple to exploit spatial correlation (i.e. correlations between adjacent channels). There might be more sophisticated and complex models able to exploit it, but relatively simple models seem to fail. It may be noted that an advantage of exploiting temporal correlation only is that the system remains very simple, since it has to process only one channel at a time and does not need a large memory area to store data from adjacent channels.

Models for capturing higher order time correlation

The code for one sample could be chosen according to the value of several previous samples instead of only the adjacent; however, even for samples which are two time bins apart the temporal correlation resulted very low, so that the improvement is so low that the gain achieved is not worth the increase in complexity of the model given by the increased number and size of codes.

Correlation between bunch area and sample values

Let us define as *area of the bunch* the energy of the bunch, i.e. the sum of its samples. It appeared that pulses with same area and length have similar

shapes; consequently, it could be possible to predict sample values from the "shape" of the bunch they belong to. More precisely, the idea was to code first the area of the bunch. Then, each sample value [3] is predicted according to the length and area of the bunch, and from its position in the bunch; finally, the prediction error of the sample value is entropy coded.

Experimental results showed that, even if a relevant gain may be obtained using this coding model, unfortunately this is not enough to compensate for the cost necessary to the coding of the area.

3 Lossy compression of TPC data

The objective of investigating lossy compression of TPC data is to explore the possibility of further increase the compression ratio and at the same time to preserve in the compressed signal the quantities that are of main interest for the experiment. These quantities are the energy, and the temporal and the spatial positions of *clusters* [1], where a cluster is a group of non-zero samples which are temporally and spatially adjacent, so that they belong to adjacent time-bins or pads in the same pad row. In other words, one cluster is made of several bunches in adjacent pads in the same pad row. Energy and positions of clusters are the quantities by which particle trajectories inside the TPC are measured, processing the data coming from each whole row of pads (about 100 pads). Particle trajectories will be calculated off-line, at the end of the experiment, by appropriate algorithms that are presently being developed and tuned at CERN using simulated data. The main idea is therefore not to preserve the value of each single signal sample, but to code the samples with the lowest number of bits so that the energy and position of the clusters, the only quantities by which the particle trajectories are measured, are preserved. Ideally the optimal acquisition system would directly evaluate such quantities on-line and directly compress these quantities before storage. Unfortunately this approach would require data coming from one whole row. The compression system receives data from up to 4 000pads, and the order in which they are received depends on the physical connections of front–end cards to the read–out chambers and is not row-by-row. This means that, in order to apply "row–oriented" algorithms, it would be necessary to store data from all 4 000 channels before starting computation of energy and cluster positions and coding. This process would introduce delay and memory requirements to the acquisition system that cannot easily be satisfied.

Therefore, appropriate quantities have been defined which are related to the

[3] with the exception of one sample per bunch, which can be easily obtained by subtraction.

signal coming from each pad, i.e. referred to bunches, from which the energy and the temporal and the spatial position of the clusters can be successively computed. Such quantities are the area (or energy) of a bunch, defined as the sum of its samples, and the Center Of Mass (CoM), defined as its temporal position. In the post-processing phase the energy of the cluster can be calculated as sum of the energies of the bunches it contains and its position as weighted average of the same bunches.

It has to be noticed that the area and the CoM of the bunches are coded losslessly; this means that, even if the compression does not preserve the single values of the samples, it does preserve these quantities without any loss. To be more precise only a quantization error for CoMs remains present, but such error can be reduced below the noise error of the original signal.

3.1 Area of bunches

The area of a bunch is simply evaluated as the sum of the values of its samples. Direct coding on probability distribution is applied to the value obtained.

3.2 Center of mass of bunches

The position of the CoM of a bunch is evaluated as the average temporal position of its samples, i.e. $t_{CoM} = (\sum_i s_i t_i) / (\sum_i s_i)$, where s_i and t_i are the values and the temporal positions of the samples of the bunch, respectively. Similarly to what has been done with the temporal information in the lossless approach, CoM positions are coded differentially, i.e. their values are substituted by the distances between CoMs of consecutive bunches. However, the distance information can not be coded *as is*, i.e. without quantization; in fact, the number of possible values, although finite, is very large, so that the entropy of such source is very large. Moreover, direct coding of the exact CoMs differential positions is also useless in practice because such precision gives an error which is far below the intrinsic noise of the original signal. Consequently, a quantization step is appropriate before coding, in order to reduce the number of possible values that CoMs can assume, and consequently to reduce the entropy.

Obviously, quantization can be applied with different quantization steps; by increasing the resolution, i.e. decreasing the quantization interval, the quantization error decreases while the entropy increases. In order to set the quantization step, it makes sense to impose a quantization error comparable with the error which is intrinsic to the data. The CoM, in fact, is affected by an error that is due to the quantization of the samples performed by the analog-

to-digital converters (ADCs) in the very first stages of the acquisition chain. As shown in appendix A, the mean square error on the position of the CoM introduced by the quantization of the samples is

$$\overline{\sigma^2_{CoM}} = \sigma^2_\varepsilon \sum_{\mathcal{B}} \left(\frac{\sum_i (t_i - t_{CoM})^2}{A^2} \right) \Big/ \mathcal{C}(\mathcal{B}) \tag{1}$$

where $\mathcal{B}$ is the set of bunches being considered, $\mathcal{C}(\mathcal{B})$ its cardinality, t_i the temporal position of sample i in the bunch, t_{CoM} the CoM of the bunch, A its area, and σ^2_ε the mean square error on the sample values (equal to 1/12). For the simulated data [4] used in this study it may be found that

$$\sqrt{\overline{\sigma^2_{CoM}}} = 0.0129 \cdot T_s$$

In table 1 this error is compared with the error introduced by the CoM quantization. It may be seen that for a quantization interval equal to $T_s/32$, where T_s is the width of a time-bin, i.e. the sampling period of the ADC, the error due to the CoM quantization is lower than the error due to the ADC quantization. Consequently, a resolution equal to $T_s/32$ has been chosen; however, it has to be added that bigger quantization intervals, in particular, $T_s/4$ and T_s, have also been considered in the performance estimations, due to the moderate increase in the error with respect to a significant decrease in bit rate.

The two parameters considered, i.e. area and CoM, provide a good description of the bunches in the case of *simple* bunches. However, such description is not sufficiently detailed for bunches which result from the superposition of two traces; in this case, two temporally close pulses are registered by a pad, with their tails overlapping, so that they are represented by a single bunch in the ALTRO data format. In this case it is necessary, before evaluating area and CoM of these bunches, to separate the contributions due to the different traces.

Rigorously, this operation should be performed exactly in the same way as it will be done by the above mentioned off-line clustering algorithms. However, since these algorithms are not completely defined yet, in the present study a simple technique has been used since the focus here is on the possibility of higher compressions rate; it is likely in fact that specific splitting algorithms do not imply relevant changes in the entropy of the quantity to code. Specifically, each bunch having a relative minimum is "cut" in correspondence with the minimum (Fig. 7); two new bunches are then obtained, and the sample in the intermediate position, i.e. the one of the local minimum, is divided by two, assigning half of it to each bunch. In terms of compression performances, this approach, though simple, should yield results very close to those provided by the future, more sophisticated, algorithms.

3.2.1 Reconstruction of the TPC signal

The compressed signal, represented, as already mentioned, by the area and the CoM of the bunches, has to be decompressed in form of a standard sampled signal to be transparent with the standard off-line processing stage, i.e. to be taken back to the ALTRO data format, in order to be processed by the off-line algorithms (which are, in fact, designed to process this kind of data). A procedure has been developed which, starting from the area and the CoM of the bunch, reconstructs samples; obviously some errors are introduced on their values if these are compared to the original signal. The precise reconstruction procedure is described below.

The objective is to capture the average shape of pulses so as to minimize the sample by sample reconstruction error. All the bunches are classified according to the {area, CoM} couple [4] and aligned at their CoM, and the mean value of the samples is computed for each time-bin. Thus, such operation defines a sort of "average bunch" (AB). Obviously, this AB has the same area and CoM, because the computation of the mean value does not alter these parameters. Moreover, it may be noted that this operation minimizes the mean square error between the actual and the mean samples. What is obtained is a "library" of ABs, each characterized by a different {area, CoM} couple.

The procedure then associates to each {area, CoM} couple the corresponding AB, and re-builds the temporal sequence of samples using the samples of the ABs. In this way, the reconstructed signal, though having possibly different samples with respect to those of the original one, is made of bunches having same area and CoM of those of the original signal, and with minimum error between each original and reconstructed sample (figure 8).

4 Simulation results

In this section, simulation results related to the described compression algorithms are reported. In the first set of simulations, the bit rate after compression is estimated by evaluating the entropy of the quantities to be coded. Then, the actual measure of the data volume is performed on one of the described lossless compression techniques, using arithmetic coding as entropy coder. Finally, these performances are compared with those given by a standard lossless compression algorithm such as *gzip* to evaluate the gain in compression

[4] Actually, only the decimal part of the CoM is considered. The integral part, in fact, does not carry any information about the sample values, i.e. the shape of the bunch, but refers only to the global shift of the bunch itself.

efficiency achieved using appropriate models instead of existing standard techniques.

4.1 Estimation of compression factors using entropy measures

The results for the lossless compression are shown in figure 9. The first column (ALTRO) represents the original volume of data in the ALTRO data format. The second column (EC) shows how this data volume is reduced if entropy coding is applied to the three fields of the ALTRO data format. The third one (SP) refers to the method which uses three different codes to code samples according to their position in the bunch (Sec. 2.1.1). Finally, the fourth column (TC) reports results of the approach that exploits temporal correlation, using 20 codes according to the position and the value of the preceding sample (Sec. 2.1.2). It may be noticed that the latter provides a compression of data to 49.2% of original size.

In figure 10 the results for the lossy technique, with several quantization levels, are compared with the ALTRO data format and the 20 code TC lossless technique considered above. It may be noticed that with the quantization level equal to $T_s/32$ the volume of data is reduced to 34.2% of its original size.

4.2 Measured compression factors using arithmetic coding

The compression factors presented in the previous paragraphs are estimated by measuring the entropy of the quantities that have to be coded. In practice, using a real coding system, performances might be worse than these estimates depending on the efficiency of the adopted entropy coder. So as to evaluate the actual performances of the 3-code SP lossless technique presented in subsection 2.1.1, the arithmetic coder presented in [6] has been applied, with a fixed probability model of the symbol source. The resulting bit rate is 30.6 bits/bunch, which is, as expected, very close to the estimated 30.3, obtained using entropy.

Similar results are expected implementing the arithmetic coder for the TC technique (subsection 2.1.2); also in this case, a slightly higher bit rate would probably result with respect to the entropy value.

To provide a comparison with a standard lossless compression tool, the same data considered above have been compressed with *gzip*, which is an implementation of the well-known Lempel-Ziv algorithm [7]. With this tool, using the highest level of compression (i.e., option "-9") the data is reduced to only 81.5% of its original size. In fact, the Lempel-Ziv algorithm performs well when strings or substrings repeat frequently in the data stream to be coded, as for example happens in text files; this, claearly, is not the case for TPC data.These results show that blind techniques, i.e. techniques that do not require explicit modeling of the source, are not efficient for our data.

5 Complexity of the compression algorithm

Finally to evaluate the feasibility of the real-time implementation of one of the lossless techniques proposed, namely the TC one, the number of operations per symbol and per second that have to be executed by the compression system have been estimated (see Table 2). Such operations mainly consist in a code selection step according to the type of symbol (bunch length, time information, or sample value), to the sample position, and to the previous sample value; the computations for the arithmetic coder considered above have also been taken into account. The number of operations per second is evaluated by assuming the worst case, in which one compression system will have to process up to 4 000 channels and 0.28 GSymbols per second. In table 3 the same results are shown for the lossy algorithm. The frequency distribution tables will need 40 kBytes of memory for the lossless technique and 4 kByte for the lossy one. Operations will be performed in 32 bit precision arithmetic.

The numbers obtained show that the described compression system can be easily implemented in real-time, either on DSPs, field programmable gate arrays, or ASICs. It may be worth noticing that even lower operation counts can be obtained by using more sophisticated arithmetic coders, such as those presented in [8,9], which do not need multiplications nor divisions but only additions and shifts.

6 Conclusions

Lossless and lossy compression approaches for the data generated by the TPC chamber in the ALICE experiment have been investigated. For the lossless technique all the samples are entropy coded using models that also can capture

temporal correlation. For the lossy approach the main idea is to preserve the two quantities that are of interest for the experiment (i.e. area and center of mass of bunches are coded without loss). The lossless approach allows a data compression to 49.3% of their original size using low complexity models such that real time implementations on low cost electronic is possible. The lossy approach achieves further compression rates in the range from 34.2% down to 23.7% accepting a quantization noise on the CoM position and errors on the sample by sample values, that in principle do not affect the results on physical quantities of interest for the experiment.

In both cases the compression algorithm can be implemented using an arithmetic coder; the overall computational complexity turns out to be reasonable, so that a real-time implementation of the system on off-the-shelf electronic devices or on simple ASICs is feasible.

A Appendix: Error on the evaluation of the center of mass of a bunch due to sample quantization

In section 3, the temporal resolution of the quantized CoMs according to the estimate of the intrinsic error in the data is derived. The samples, indeed, are affected by the error due to the quantization made by the ADCs in the very first stages of the acquisition chain; this error, obviously, propagates in the calculation of the CoM. As an example, in figure 11 it is shown how the quantization error on the samples propagates on the CoM in the case of a 3-sample bunch. It makes sense to increase the resolution of the CoM quantization up to the point where the two errors are of comparable energy.

The aim of this appendix is to quantify this error by evaluating its mean value and standard deviation. Since these quantities depend on the bunch shape, the average are evaluated on all bunches of the simulated data used in the rest of this work.

Let s_i be the non-quantized sample values, $\hat{s}_i = s_i + \varepsilon_i$ the quantized ones and t_i the relative instants of time. Let us suppose that the quantization error ε_i of each sample be uniformly distributed in the interval $(-0.5, +0.5]$ and independent on the errors on the other samples. Then

$$
\begin{aligned}
E[\varepsilon_i] &= 0 \\
\sigma_\varepsilon^2 &= 1/12 \\
E[\varepsilon_i \varepsilon_j] &= \begin{cases} \sigma_\varepsilon^2 & i = j \\ E[\varepsilon_i]\, E[\varepsilon_j] = 0 & i \neq j \end{cases}
\end{aligned}
$$

that is, denoting with $\delta_{i,j}$ the Kronecker symbol,

$$E[\varepsilon_i \varepsilon_j] = \sigma_\varepsilon^2 \, \delta_{i,j}$$

The CoM calculated with the non-quantized values is:

$$t_{CoM} = \frac{\sum_i s_i \, t_i}{A}$$

where $A = \sum_i s_i$ is the area of the bunch; in turn, using the quantized values, the CoM is:

$$\hat{t}_{CoM} = \frac{\sum_i \hat{s}_i \, t_i}{\hat{A}}$$

where $\hat{A} = \sum_i \hat{s}_i$. For t_{CoM} the partial derivatives with respect to s_i are

$$\begin{aligned} \frac{\partial \, t_{CoM}}{\partial s_i} &= \frac{\partial}{\partial s_i} \left(\frac{\sum_j s_j \, t_j}{\sum_j s_j} \right) \\ &= \frac{t_i}{\sum_j s_j} - \frac{\sum_j s_j \, t_j}{\left(\sum_j s_j \right)^2} \end{aligned}$$

and the total differential is

$$\mathrm{d}t_{CoM} = \sum_i \frac{\partial \, t_{CoM}}{\partial s_i} \, \mathrm{d}s_i$$

It follows that the error on the CoM due to the sample quantization is

$$\begin{aligned} \varepsilon_{CoM} &= \sum_i \frac{\partial \, t_{CoM}}{\partial s_i} \, \varepsilon_i \\ &= \sum_i \left(\frac{t_i}{\sum_j s_j} - \frac{\sum_j s_j \, t_j}{\left(\sum_j s_j \right)^2} \right) \varepsilon_i \\ &= \frac{\sum_i \left(t_i - t_{CoM} \right) \varepsilon_i}{A} \end{aligned}$$

Its statistics is described by:

$$E[\varepsilon_{CoM}] = E\left[\frac{\sum_i \varepsilon_i \left(t_i - t_{CoM} \right)}{A} \right] = \frac{\sum_i E[\varepsilon_i] \left(t_i - t_{CoM} \right)}{A} = 0$$

$$
\begin{aligned}
\sigma^2_{CoM} &= E[\varepsilon^2_{CoM}] \\
&= E\left[\frac{\left(\sum_i \varepsilon_i \left(t_i - t_{CoM}\right)\right)^2}{A^2}\right] \\
&= \frac{\sum_{i,j} E[\varepsilon_i \varepsilon_j] \left(t_i - t_{CoM}\right)\left(t_j - t_{CoM}\right)}{A^2} \\
&= \sigma^2_\varepsilon \frac{\sum_i \left(t_i - t_{CoM}\right)^2}{A^2} \\
\max(\varepsilon_{CoM}) &= \frac{\sum_i 0.5 \left|t_i - t_{CoM}\right|}{A} = 0.5 \frac{\sum_i \left|t_i - t_{CoM}\right|}{A}
\end{aligned}
$$

By averaging these quantities on a set of bunches $\mathcal{B}$ with cardinality $\mathcal{C}(\mathcal{B})$, we get

$$
\begin{aligned}
\overline{E[\varepsilon_{CoM}]} &= 0 \\
\overline{\sigma^2_{CoM}} &= \sigma^2_\varepsilon \sum_{\mathcal{B}} \left(\frac{\sum_i \left(t_i - t_{CoM}\right)^2}{A^2}\right) \Big/ \mathcal{C}(\mathcal{B}) \\
\overline{\max(\varepsilon_{CoM})} &= 0.5 \sum_{\mathcal{B}} \left(\frac{\sum_i \left|t_i - t_{CoM}\right|}{A}\right) \Big/ \mathcal{C}(\mathcal{B})
\end{aligned}
$$

By evaluating the sums on the set of simulated bunches $\mathcal{B}$ provided by CERN in [4], it results

$$
\begin{aligned}
\overline{E[\varepsilon_{CoM}]} &= 0 \\
\sqrt{\overline{\sigma^2_{CoM}}} &= 0.0129 \cdot T_s \\
\overline{\max(\varepsilon_{CoM})} &= 0.0293 \cdot T_s
\end{aligned}
$$

References

[1] ALICE Collaboration, ALICE TDR 7, ALICE Technical Design Report of the Time Projection Chamber, Tech. Rep. CERN/LHCC/2000–001, CERN, http://alice.web.cern.ch/Alice/TDR/TPC/alice_tpc.pdf (January 2000).

[2] ALICE – A Large Ion Collider Experiment, http://alice.web.cern.ch/Alice.

[3] L. Musa, Electronics status, Presentation at the ALICE Week (28 May – 1 June 2001).

[4] V. C. de Verdiere, TPC Row binary files, http://AliSoft.cern.ch/ ˜vcolin/TPC/g.bin.gz, Simulated data for 12 chambers and 446 time-bins.

[5] A. Nicolaucig, Compressione dei dati generati dalla Time Projection Chamber nell'esperimento ALICE presso il CERN, Master's thesis, Università di Trieste (2000).

[6] I. H. Witten, R. M. Neal, J. G. Cleary, Arithmetic coding for data compression, Communications of the ACM 30 (6) (1987) 520–540.

[7] J. Ziv, A. Lempel, A universal algorithm for data compression, IEEE Transactions On Information Theory 23 (3) (1977) 337–343.

[8] J. Rissanen, K. M. Mohiuddin, A multiplication-free multialphabet arithmetic code, IEEE Transactions on Communications 37 (2).

[9] A. Moffat, R. Neal, I. H. Witten, Arithmetic coding revisited, ACM Transactions on Information Systems 16 (3) (1998) 256–294.

List of Figures

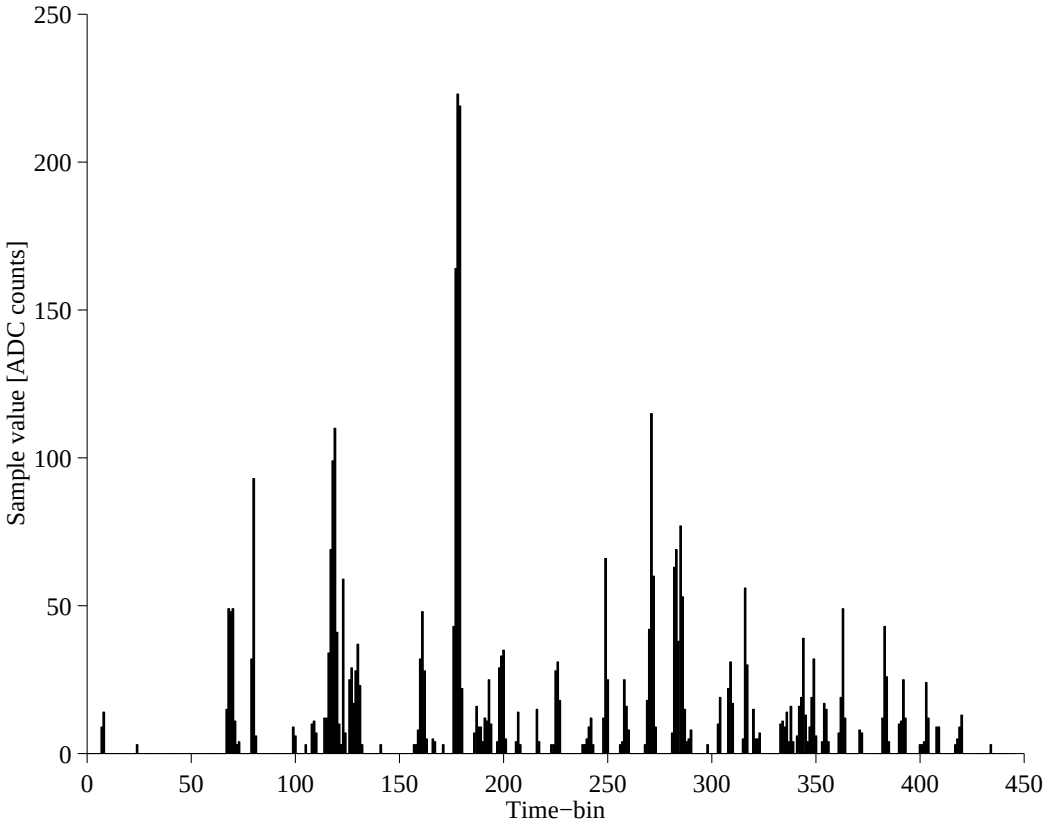

Fig. 1. Simulated TPC data (according to [4]) showing the possible temporal behaviour of the signal on a TPC pad.

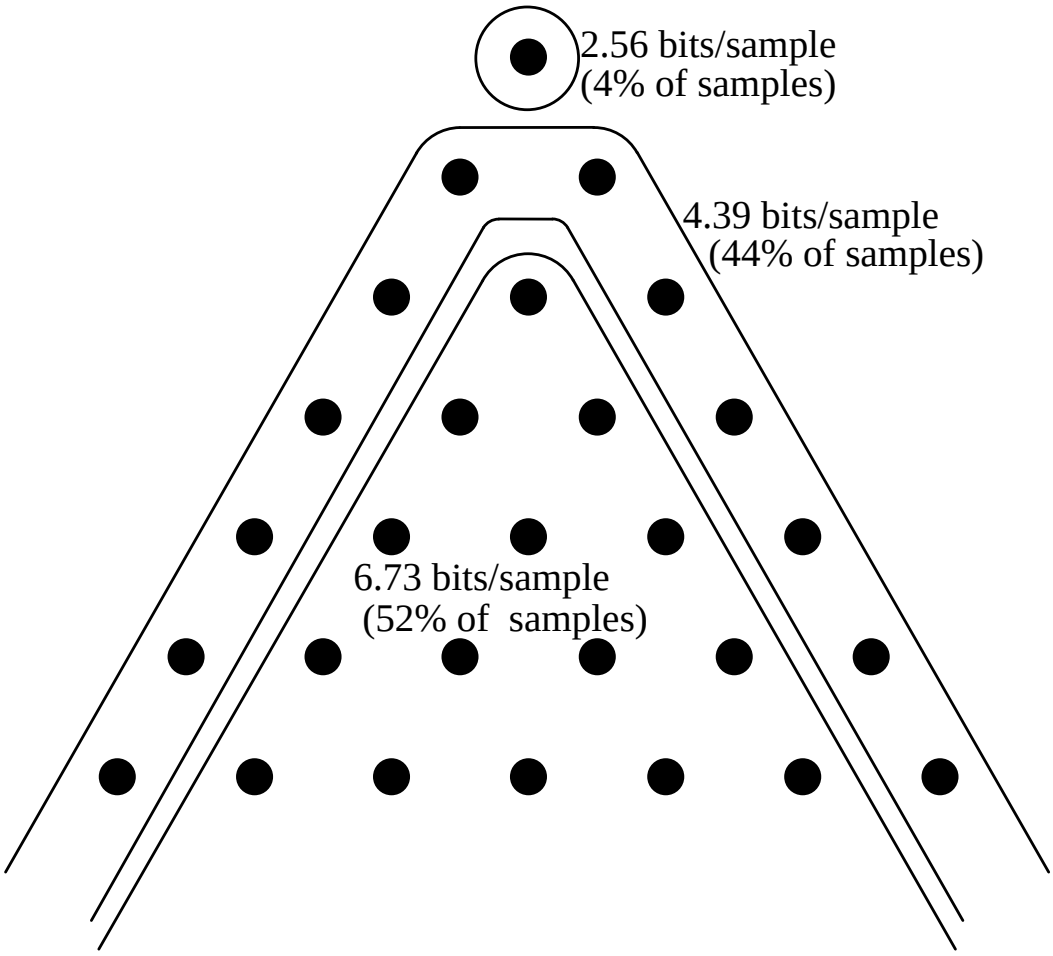

Fig. 2. Classification of samples according to their position in the bunch. In the figure, different rows refer to bunches having different length. Three classes are used: isolated sample, first or last sample, and sample in intermediate position. The entropy and the percentage of the samples belonging to each of the three classes is also reported; the average entropy of this model is 5.59 bits per sample.

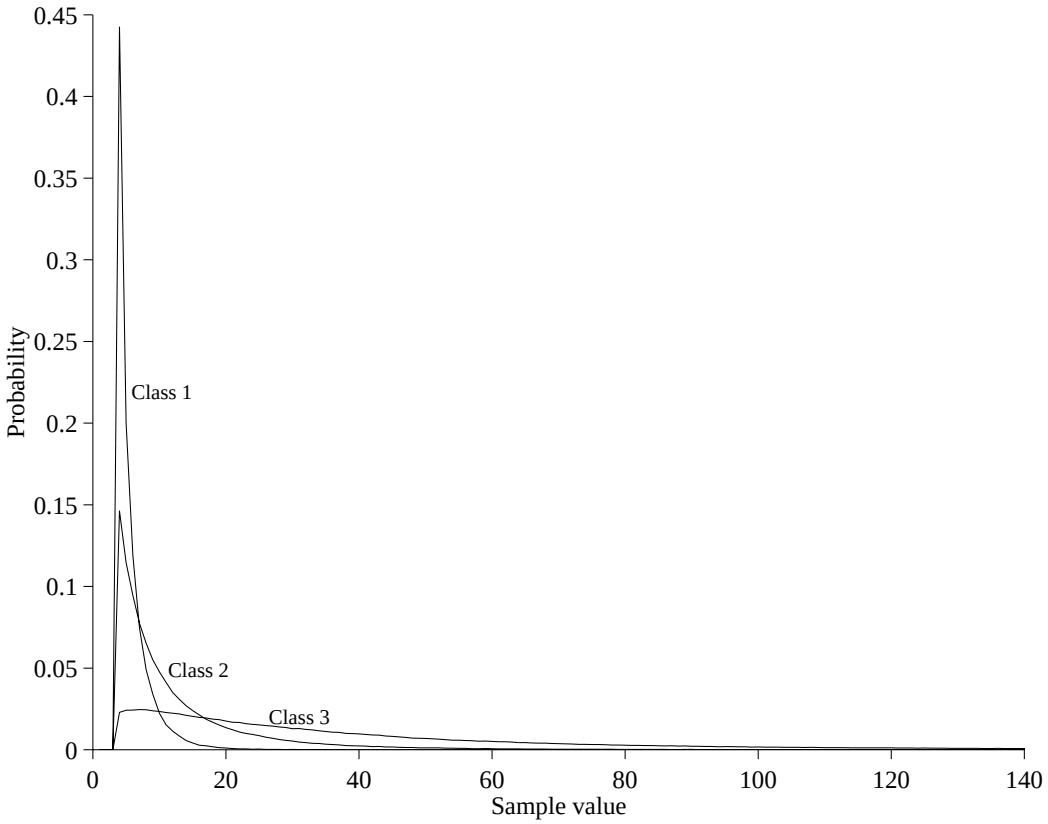

Fig. 3. Probability distributions for the samples belonging to the 3 classes shown in Fig. 2

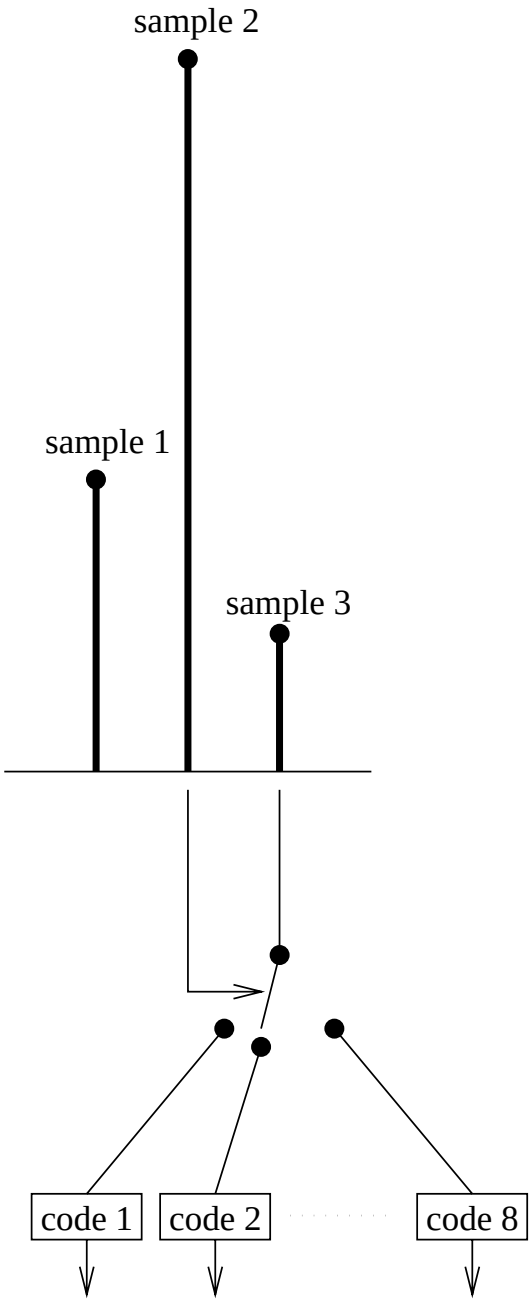

Fig. 4. Example of conditioned probability model exploiting temporal correlation between sample 2 and sample 3 in a three sample bunch. One out of eight different codes is chosen for sample 3, according to the value of sample 2.

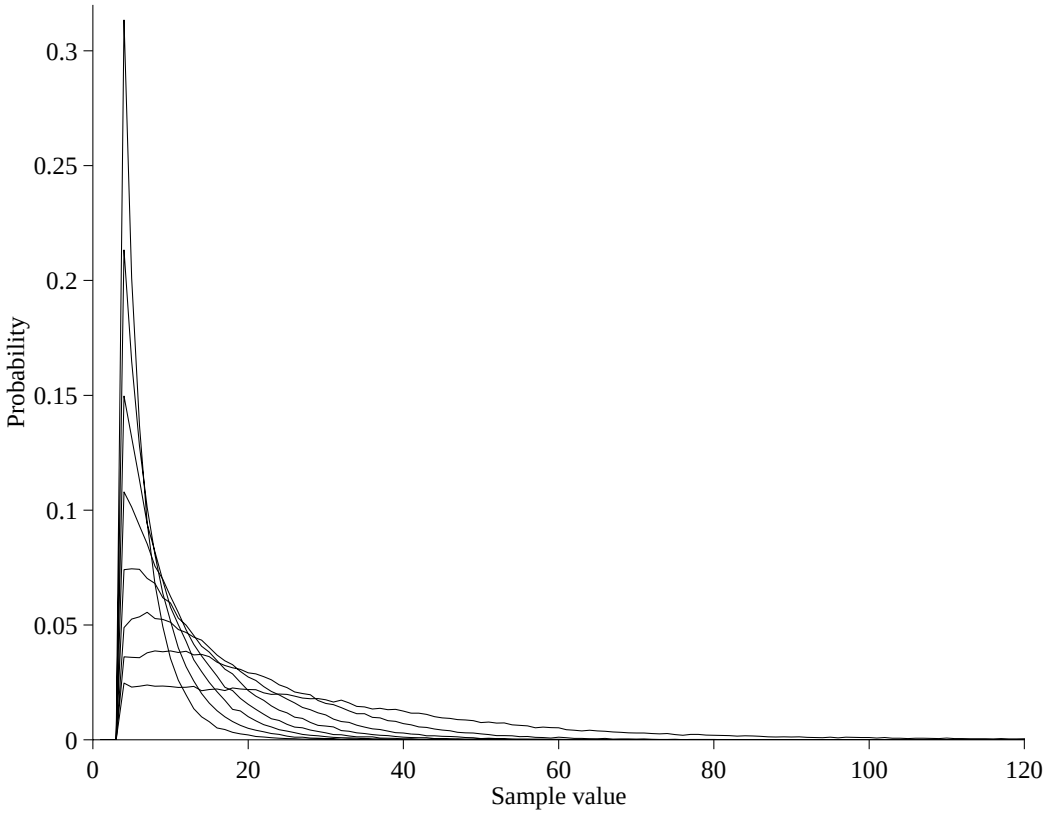

Fig. 5. Distributions of sample 3 according to the value of sample 2, or, more precisely, to the interval (one out of 8) to which sample 2 belongs (see Fig. 4).

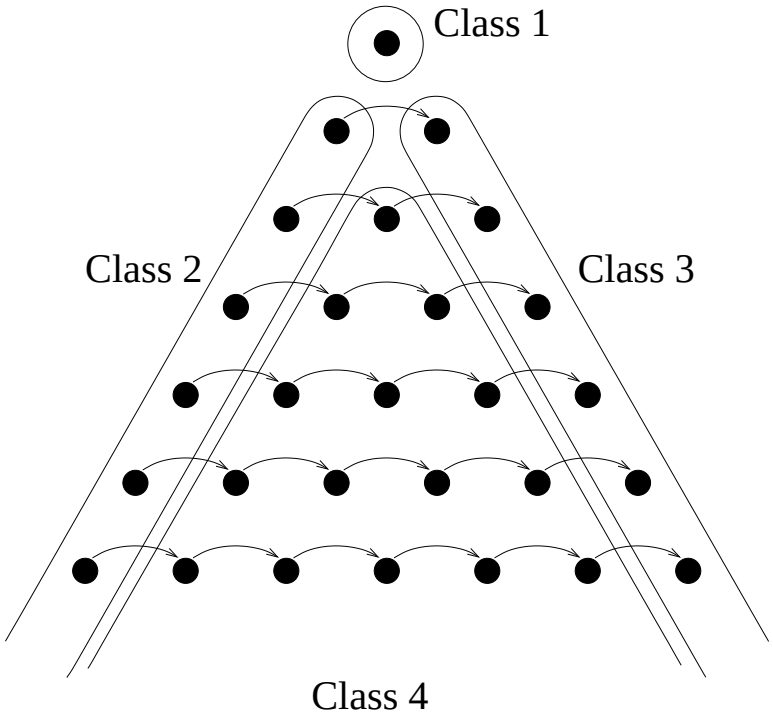

Fig. 6. Classification of samples exploiting both position in the bunch and temporal correlation. As in Fig. 2, different rows refer to bunches having different length. A total of 20 codes is used: one each for classes 1 and 2, eight for class 3, eight for class 4, and two for bunch length and time information.

Fig. 7. Example of "cut" of a bunch originated by two temporally close traces. A 5-sample bunch is cut into two 3-sample ones.

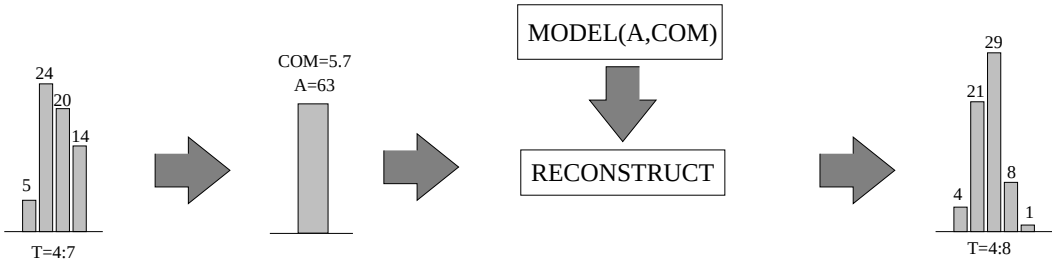

Fig. 8. Example of signal reconstruction for a bunch having area equal to 63 and decimal part of the CoM equal to 0.7. A 5-samples bunch is generated.

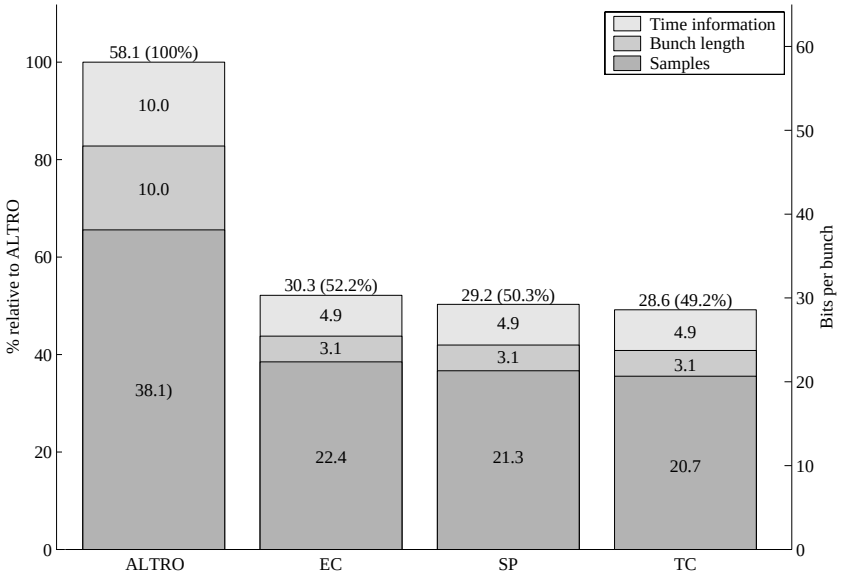

Fig. 9. Performances of several lossless techniques compared to the zero suppressed ALTRO data format. ALTRO: original ALTRO data; EC: entropy coding of sample values, bunch length, and time information; SP: classification of samples according to their position (3 codes used); TC: coding technique that exploits temporal correlation (20 codes used).
Numbers in the columns represent the number of bits per bunch dedicated to each field; numbers on top of the columns represent the overall number of bits per bunch and, in parenthesis, the size with respect to the original ALTRO data format.

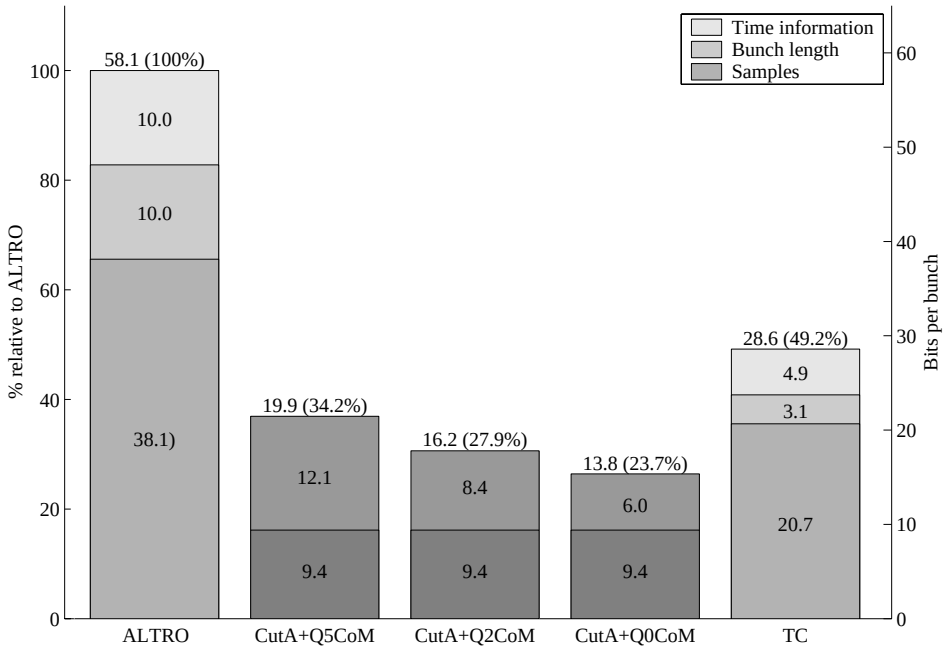

Fig. 10. Lossy techniques performances. ALTRO: original ALTRO data; CutA+Q5CoM: proposed lossy coding, with CoM quantization at $T_s/32$; CutA+Q2CoM: the same, with a coarser CoM quantization ($T_s/4$); CutA+Q0CoM: the same, with T_s. For a comparison, also the results of the 20-code lossless compression technique TC is reported; the meaning of the various numbers is the same as in the previous picture

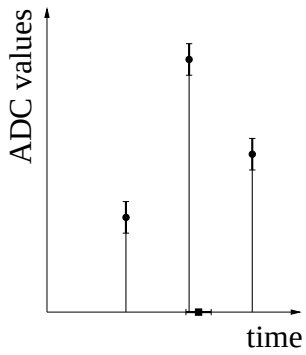

Fig. 11. Propagation of the ADC quantization error on the CoM

List of Tables

Quant. step/T_s	bits/bunch	mean/T_s	σ/T_s	max/T_s	Rel. no of zeros (%)
1	5.1	$-1.17\cdot10^{-2}$	0.2758	0.4998	17.94
1/2	6.1	$-2.4\cdot10^{-3}$	0.1291	0.2499	20.28
1/4	7.1	$-1.9\cdot10^{-3}$	0.0658	0.1249	21.24
1/8	8.1	$-5.78\cdot10^{-4}$	0.0330	0.0625	22.76
1/16	9.1	$-1.86\cdot10^{-4}$	0.0161	0.0312	23.74
1/32	10.1	$-5.28\cdot10^{-5}$	0.0081	0.0156	24.29
ADC quantization only		0	0.0129	0.0293	—

Table 1
Mean value, standard deviation and maximum value of the quantization error of the CoM position for different resolutions and corresponding bit rates, compared to the intrinsic error due to the ADC quantization (bottom row). The last column represents the percentage of bunches for which the quantization error is zero.

	sums	multiplications	divisions	jumps	memory accesses
Time correlation (op/symb.)	10	–	–	8	1
Arithmetic coding (op/symb.)	93	12	2	15	4
Total (op/symb.)	103	12	2	23	5
Time correlation (op/s)	2.8 G	–	–	2.2 G	0.28 G
Arithmetic coding (op/s)	26 G	3.2 G	0.55 G	4.0 G	1.1 G
Total (op/s)	28 G	3.2 G	0.55 G	6.2 G	1.4 G

Table 2
Evaluation of the complexity of the algorithm for lossless compression. The number of operations per second refers to the worst case processing of upto 4 kchannels and 0.28 GByte/s.

	sums	multipli-cations	divisions	jumps	memory accesses
Area and CoM (op/symb.)	1.9	0.27	0.21	0.48	0
Arith. coding (op/symb.)	69	8.2	0.85	11	1.7
Total (op/symb.)	71	8.5	1.1	12	1.7
Area and CoM (op/s)	516 M	75 M	58 M	133 M	0
Arith. coding (op/s)	19 G	2.2 G	0.23 G	3.1 G	0.47 G
Total (op/s)	20 G	2.3 G	0.29 G	3.2 G	0.47 G

Table 3
Evaluation of the complexity of the algorithm for lossy compression. The number of operations per second refers to the worst case processing of up to 4 kchannels and 0.28 GByte/s.